**이번 주 진행 상황**

1. 회귀에 대해서 공부
2. Sckit 라이브러리로 회귀 곡선을 긋는 x, y 값들의 좌표 생성
3. D3 코드에 적용시켜 화면 구현 – 신뢰구간 구현은 방법을 찾는 중
4. 비선형 그래프에 대한 공부 중 (가우시안, 포아송에 대한 개념 등)

**구현 현황**

X, y 좌표 및 regression 값들을 반환 하는 함수 작성

def quadratic\_y\_values(df, x\_name, y\_name, degree=2):

    X = df[[x\_name]].values

    Y = df[y\_name].values

    lr = LinearRegression()

    quadratic = PolynomialFeatures(degree=degree)

    X\_quad = quadratic.fit\_transform(X)

    X\_fit = np.arange(0, X.max()+1, 100)[:, np.newaxis]

    lr.fit(X\_quad, Y)

    y\_quad\_fit = lr.predict(quadratic.fit\_transform(X\_fit))

    return df[x\_name].values, Y, X\_fit, y\_quad\_fit

Views에서 D3.js에서 사용할 수 있도록 데이터 변환

# api/scatter

# d3 data type

def scatter\_plot(request):

    print("scatter\_plot")

    # connect to mongodb

    mc = mongo\_connect.mongodb()

    mc.set\_db('ceramicdb')

    mc.set\_collection('ceramic')

    col = mc.get\_collection()

    # query

    result = col.find({})

    # 데이터 프레임으로 전환

    df = pd.DataFrame(result)

    input\_name = 'Gap'

    output\_name = 'average\_thickness'

    # 문자열을 숫자로 바꾸고 NAN 행 제거

    df['average\_thickness'] = pd.to\_numeric(df['average\_thickness'], errors='coerce')

    df = df.dropna(axis = 0)

    index = list(df['index'].values)

    # x, y scatter 값과 회귀 곡선을 그리는 좌표 반환

    x, y, x\_fit, y\_quad = quadratic\_y\_values(df, input\_name, output\_name)

    x\_fit = list(itertools.chain(\*x\_fit))

    x = list(x)

    # D3 데이터 형식으로 넣어줄 수 있도록 형태 변환

    scatter\_len = len(x)

    line\_len = len(x\_fit)

    scatter\_data = []

    for idx in range(scatter\_len):

        temp = {

            "index" : int(index[idx]),

            "x\_data" : int(x[idx]),

            "y\_data" : int(y[idx]),

        }

        scatter\_data.append(temp)

    line\_data = []

    for idx in range(line\_len):

        temp = {

            "x\_fit" : int(x\_fit[idx]),

            "y\_quad" : int(y\_quad[idx]),

        }

        line\_data.append(temp)

    context = {

        'scatter\_data' : scatter\_data,

        'line\_data' : line\_data,

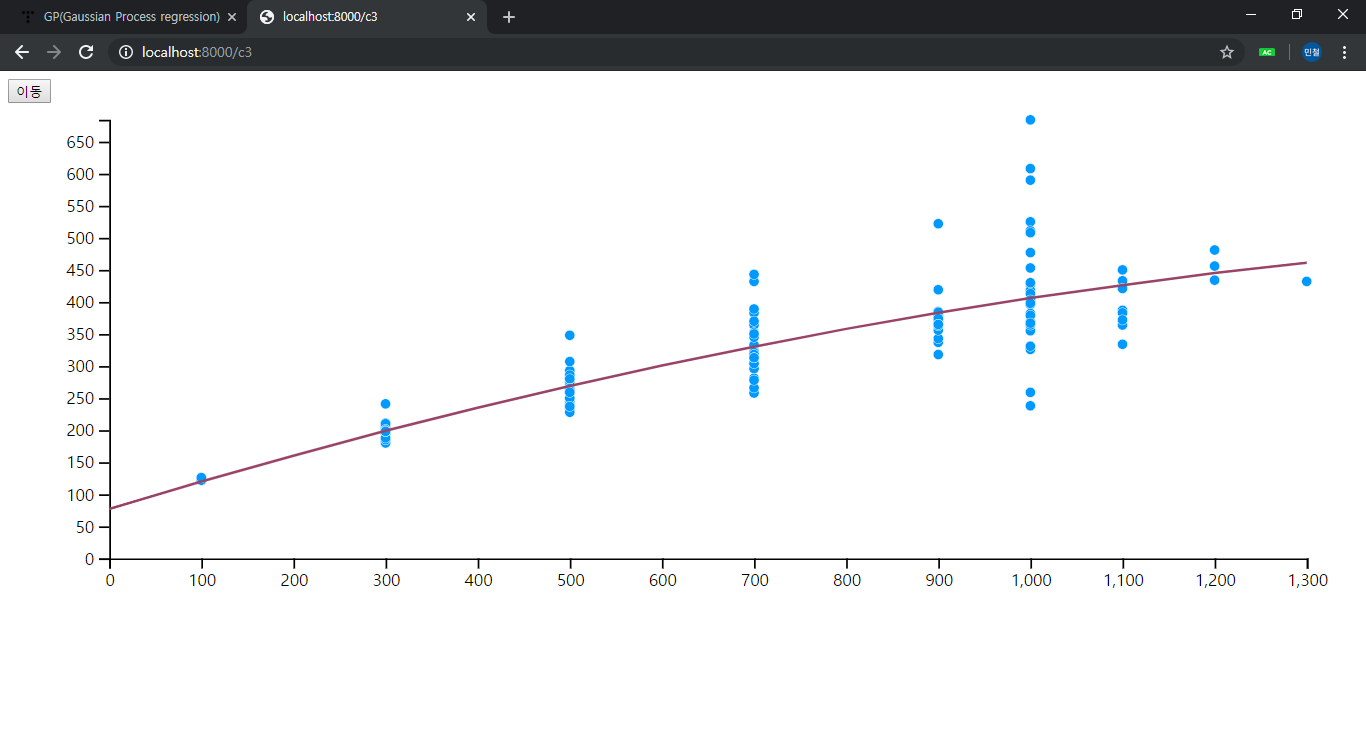
    }

    print("json sended")

    # Json으로 반환

    return JsonResponse(context)

출력화면



X축 : Gap, Y축 : average\_thickness

2차수로 계산한 linear regression

**비선형 그래프에 대한 공부**

가우시안 프로세스 (GP, Gaussian Process)

가우시안 프로세스 리그래션 : 한 차원의 변수에서, 변수간의 관계에 kernel을 정의하고, 이 커널에 따라 무한개의 가우시안을 가정한다. 이를 통해 한 차원에는 무한개의 가우시안이 존재하고 이 가우시안의 평균과 분산을 이용해 해당 차원에서의 변수를 Regression 할 수 있다.

가우시안 Regression이란 가정한 한 차원에 무한개의 가우시안이 존재하는 발상을 의미

예시 ) 1분마다 빈 택시가 한대씩 지나가는데 그 택시의 이동속도는 어떻게 될까? 를 가우시안 분포로 추정하는 것 – 그럼 1분마다 X가 나타나는데 그 두 x간의 높이가 서로 가우시안 분포를 따르는 것,

그전에 지나간 택시가 매우 빨랐다면 그 다음 택시도 빠를까? 라는 것이 GP

커널에 따른 가우시안 그래프 출력 (예시 코드)

print(\_\_doc\_\_)

# Authors: Jan Hendrik Metzen <jhm@informatik.uni-bremen.de>

#

# License: BSD 3 clause

import numpy as np

from matplotlib import pyplot as plt

from sklearn.gaussian\_process import GaussianProcessRegressor

from sklearn.gaussian\_process.kernels import (RBF, Matern, RationalQuadratic,

                                              ExpSineSquared, DotProduct,

                                              ConstantKernel)

kernels = [1.0 \* RBF(length\_scale=1.0, length\_scale\_bounds=(1e-1, 10.0)),

           1.0 \* RationalQuadratic(length\_scale=1.0, alpha=0.1),

           1.0 \* ExpSineSquared(length\_scale=1.0, periodicity=3.0,

                                length\_scale\_bounds=(0.1, 10.0),

                                periodicity\_bounds=(1.0, 10.0)),

           ConstantKernel(0.1, (0.01, 10.0))

               \* (DotProduct(sigma\_0=1.0, sigma\_0\_bounds=(0.1, 10.0)) \*\* 2),

           1.0 \* Matern(length\_scale=1.0, length\_scale\_bounds=(1e-1, 10.0),

                        nu=1.5)]

for kernel in kernels:

    # Specify Gaussian Process

    gp = GaussianProcessRegressor(kernel=kernel)

    # Plot prior

    plt.figure(figsize=(8, 8))

    plt.subplot(2, 1, 1)

    X\_ = np.linspace(0, 5, 100)

    y\_mean, y\_std = gp.predict(X\_[:, np.newaxis], return\_std=True)

    plt.plot(X\_, y\_mean, 'k', lw=3, zorder=9)

    plt.fill\_between(X\_, y\_mean - y\_std, y\_mean + y\_std,

                     alpha=0.2, color='k')

    y\_samples = gp.sample\_y(X\_[:, np.newaxis], 10)

    plt.plot(X\_, y\_samples, lw=1)

    plt.xlim(0, 5)

    plt.ylim(-3, 3)

    plt.title("Prior (kernel:  %s)" % kernel, fontsize=12)

    # Generate data and fit GP

    rng = np.random.RandomState(4)

    X = rng.uniform(0, 5, 10)[:, np.newaxis]

    y = np.sin((X[:, 0] - 2.5) \*\* 2)

    gp.fit(X, y)

    # Plot posterior

    plt.subplot(2, 1, 2)

    X\_ = np.linspace(0, 5, 100)

    y\_mean, y\_std = gp.predict(X\_[:, np.newaxis], return\_std=True)

    plt.plot(X\_, y\_mean, 'k', lw=3, zorder=9)

    plt.fill\_between(X\_, y\_mean - y\_std, y\_mean + y\_std,

                     alpha=0.2, color='k')

    y\_samples = gp.sample\_y(X\_[:, np.newaxis], 10)

    plt.plot(X\_, y\_samples, lw=1)

    plt.scatter(X[:, 0], y, c='r', s=50, zorder=10, edgecolors=(0, 0, 0))

    plt.xlim(0, 5)

    plt.ylim(-3, 3)

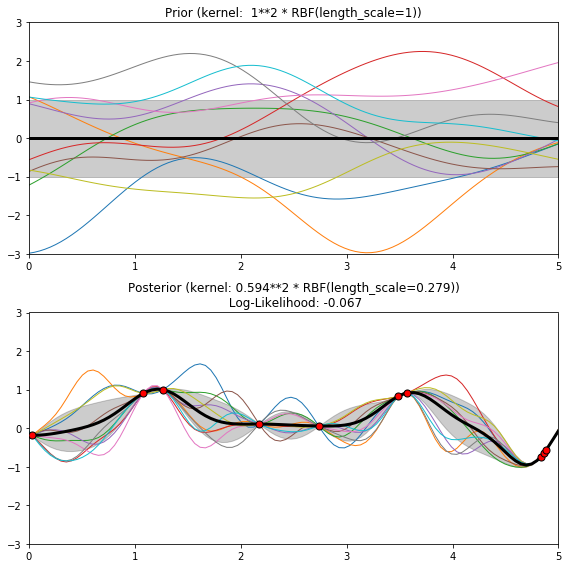
    plt.title("Posterior (kernel: %s)\n Log-Likelihood: %.3f"

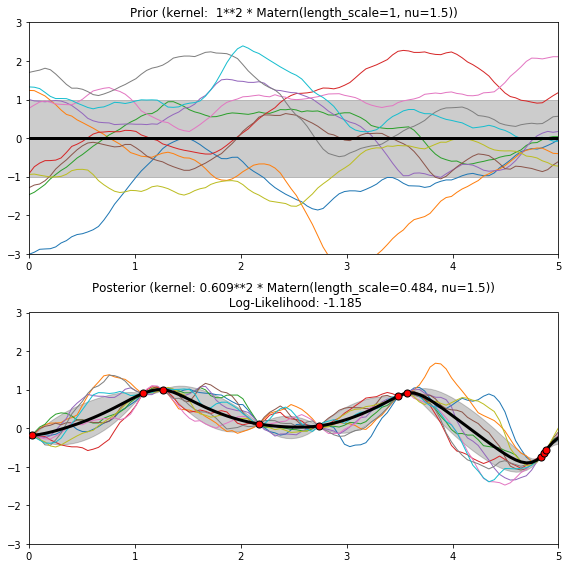
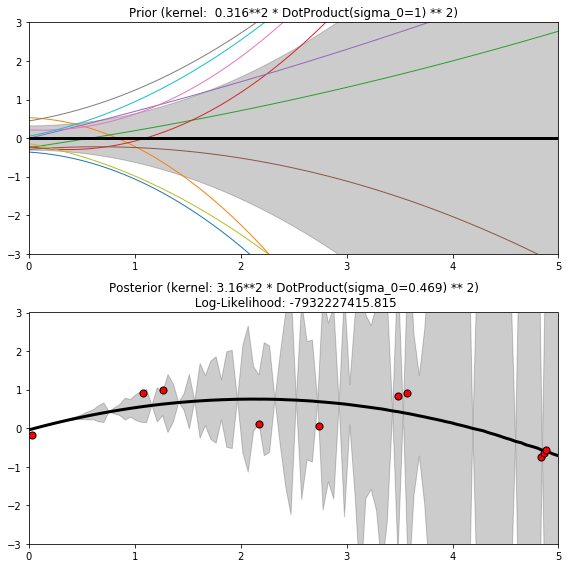
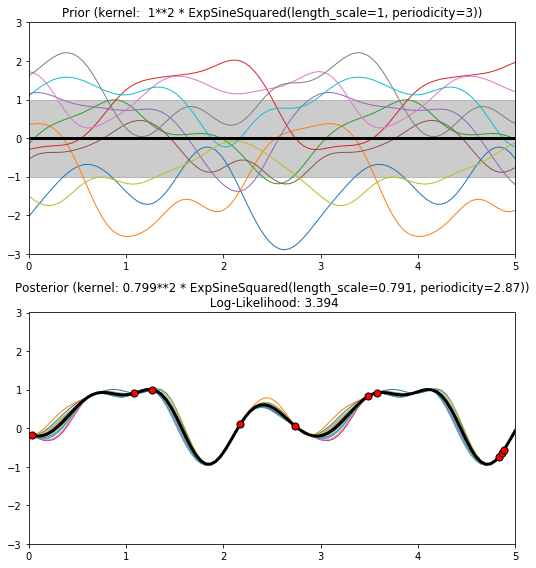
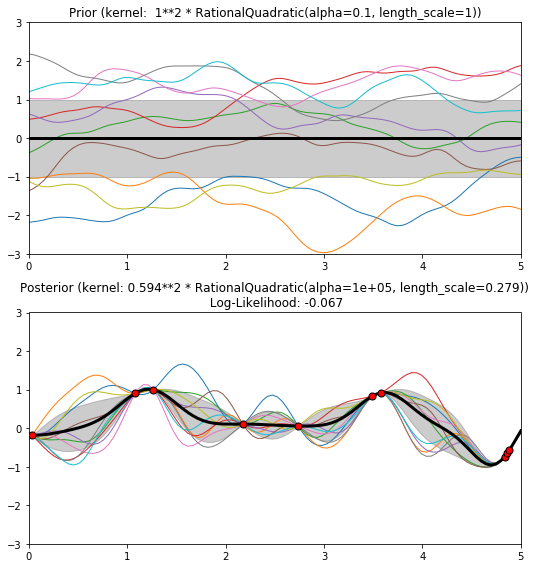
              % (gp.kernel\_, gp.log\_marginal\_likelihood(gp.kernel\_.theta)),

              fontsize=12)

    plt.tight\_layout()

plt.show()





**그 외**

선형 회귀에 대해서 조금 하고나서 다음 비선형으로 넘어가려니까 포아송이라던지 가우시안 등 같은 잘 모르는 개념들이 계속 나와 조금 어려움을 겪고 있습니다. 단순한 수학적 개념은 한글 문서로도 많은데 가우시안 프로세스 같은 파이썬 모듈들은 한국어로 된 문서가 별로 없는 것이 많아서 코드에 대한 이해가 많이 떨어집니다. 그러다 보니 좀 더 여러가지 그래프 화면을 개발하고 싶은데 더디게 되네요.

이 부분들에 대해서는 제가 조금 더 공부하고 코드를 작성해보면서 계속해보겠습니다.

그리고 지금은 무작정 상관관계, 회귀분석 화면을 구현하고 있긴 한데, 이러한 그래프들을 얼마나 어떻게 화면에 보여주는지 감이 잘 안옵니다. 사용자가 어떠한 데이터를 원하는지를 좀 알아야 화면에 대한 아이디어가 떠오를 것 같은데 조언같은 것을 해 주실 수 있을까요?